Lecture 6

Approximate Counting

- connection to uniform generation

Randomized Complexity Classes

Begin Pairwise Independence:

example Max Cut randomized algorithm

- only uses pairwise independence!

Approx counting for DNF:

Fully polynomial randomized approximation scheme (FPRAS)

Given ϕ, ε

s.t. z = # sat assignments to ϕ

Output y s.t.

$$\frac{z}{1+\varepsilon} \leq y \leq z \cdot (1+\varepsilon)$$

with prob $\geq 3/4$

Will use:

(1) uniform generation of DNF sat assignments

(2) "Downward self-reducibility" of DNF

Downward self-reducibility: (dsr)

Can compute problem by solving smaller subproblems & putting together answers via poly time computation.

Why is #-DNF dsr.?

$$\#\phi(x_1 \cdots x_n) = \#\phi(x_1 = T, x_2, \ldots x_n) +$$

$$\#\phi(x_1 = F, x_2, \ldots x_n)$$

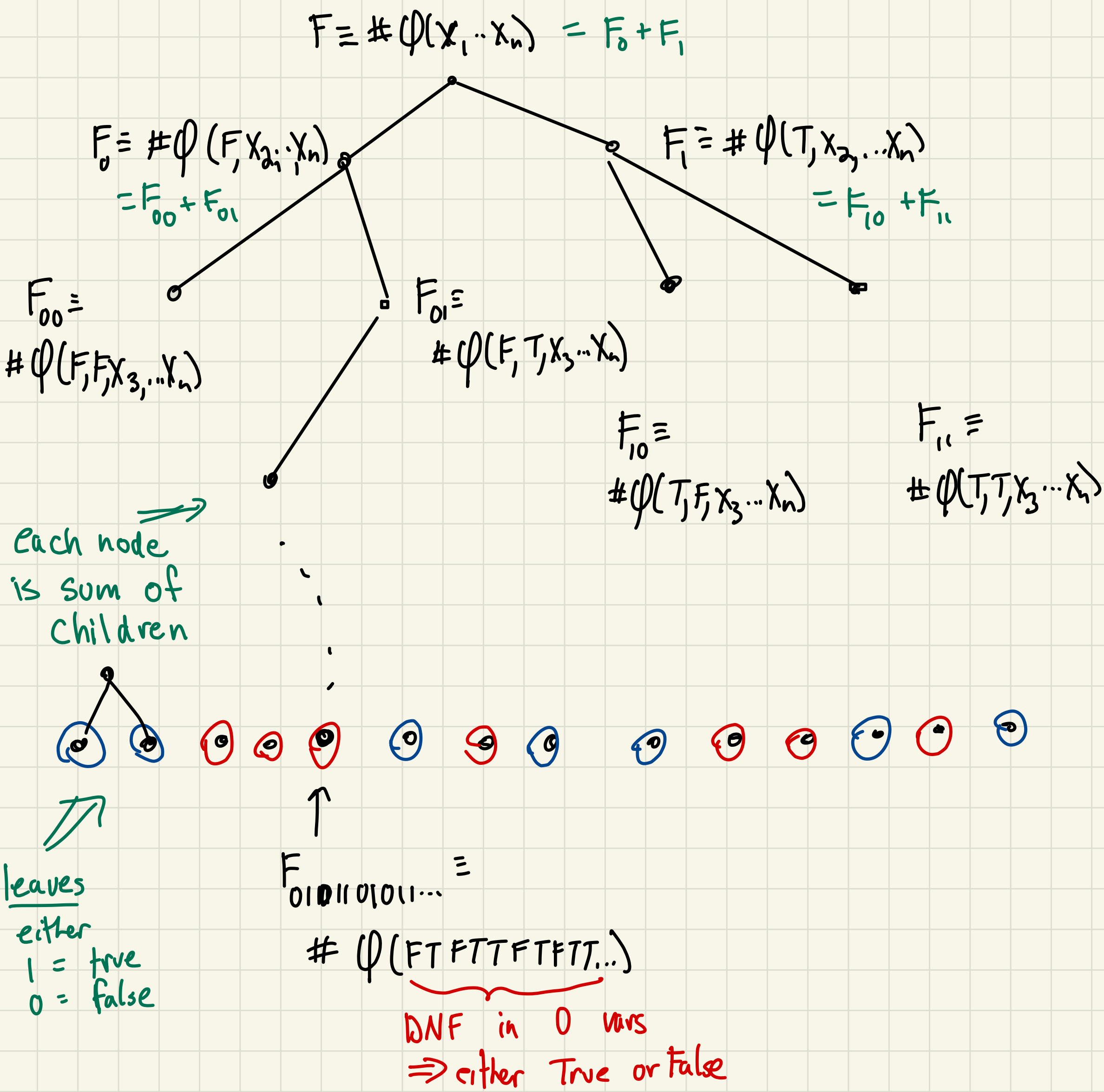
both are still DNFs but in n-1 vars.

e.g. $\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2)$

$= \#(\bar{x}_2 \vee x_2 \vee \bar{x}_2)$ ← # settings where $x_1 = T$

$+ \#(\bar{x}_2)$ ← # settings where $x_1 = F$

Downward Self-Reducibility Tree



example

$$\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2) = 3$$

$x_1 = T$:

$$\#(\bar{x}_2 \vee x_2 \vee \bar{x}_2) = 2$$

$$= \#(T)$$

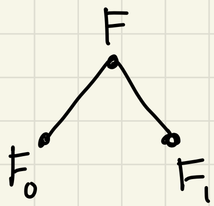
- $x_2 = T$: T, 1
- $x_2 = F$: T, 1

$x_1 = F$:

$$\#(\bar{x}_2) = 1$$

- $x_2 = T$: F, 0
- $x_2 = F$: T, 1

Approximate Counting Algorithm for #DNF



Let $S_1 = \underbrace{\dfrac{F_1}{F}}_{\text{Fraction of SAT assignments s.t. } x_1 = T} \Rightarrow F = \dfrac{F_1}{S_1}$

main insight: for DNF, we can estimate S_1 via sampling!

- uniformly generate k sat assignments ← we know how to do this for DNF!!
- $\tilde{S}_1 \leftarrow \dfrac{\#\text{ with } x_1 = T}{k}$

But how do we compute F_1?

recursively!

$$F_1 = \frac{F_{11}}{S_{11}}$$

F_{11} ← recurse

S_{11} ← estimate

So $F = \dfrac{F_{b_1}}{S_{b_1}} = \dfrac{F_{b_1 b_2}}{S_{b_1} \cdot S_{b_1 b_2}} = \dfrac{F_{b_1 b_2 b_3}}{S_{b_1} \cdot S_{b_1 b_2} \cdot S_{b_1 b_2 b_3}}$

$\vdots$

$= \dfrac{1}{\prod_{i=1}^{n} S_{b_1 \cdots b_i}}$

Potential Difficulties:

1. if $F_{b_1 \cdots b_n} = 0$ this doesn't work
2. Is approximation of $S_{b_1 \cdots b_i}$'s good enough? only get additive estimates :(

Idea Always take path of "larger" child

might guess wrong when both have lots of SAT assignments but soon will show that is ok

Claim if always pick b_i s.t. $F_{b_1 \cdots b_i} > F_{b_1 \cdots \bar{b}_i}$

then always reach SAT assignment leaf.

(so $F_{b_1 \cdots b_n} = 1$)

larger child choice of path gives:

$\boxed{\text{small additive error} \Rightarrow \text{small multiplicative error}}$

Idea estimate each $S_{b_1 \cdots b_i}$ to within $\frac{\varepsilon}{6n}$

additive error (approximation error) (using Chernoff bnds, need only poly$(\frac{2n}{\varepsilon}, \log 4n)$ samples to get confidence error $< \frac{1}{4n}$)

$\Rightarrow$ if $1 \geq r \geq 1/3$ ← hopefully $\frac{1}{2}$ but might pick wrong path

union bnd over all i to get prob of error $< \frac{1}{4}$

$$r + \frac{\varepsilon}{6n} \leq r\left(1 + \frac{\varepsilon}{6n \cdot r}\right) \leq r\left(1 + \frac{\varepsilon}{2n}\right)$$

* slight issue: might be estimating $1-r$ if pick wrong path. We will ignore this for now.

$$r - \frac{\varepsilon}{6n} \geq r\left(1 - \frac{\varepsilon}{6nr}\right) \geq r\left(1 - \frac{\varepsilon}{6n}\right)$$

Claim

$$\text{output} \leq \frac{F_{b_1}}{\tilde{S}_{b_1}} \leq \frac{F_{b_1 b_2}}{\tilde{S}_{b_1} \tilde{S}_{b_1 b_2}} \leq \cdots \leq \frac{1}{\prod \tilde{S}_{b_1 \cdots b_i}}$$

$$\leq \frac{\left(1 + \frac{\varepsilon}{3n}\right)^n}{\prod S_{b_1 \cdots b_n}} = F \cdot \underbrace{\left(1 + \frac{\varepsilon}{3n}\right)^n}_{1 + \frac{\varepsilon}{3} + \frac{(\frac{\varepsilon}{3})^2}{2!} + \cdots \leq 1 + \frac{\varepsilon}{2}} \leq F\left(1 + \frac{\varepsilon}{2}\right)$$

Taylor series

similarly, output $\geq \frac{F}{1+\varepsilon}$

■

Recursive Algorithm

- estimate S_0, S_1 from unif generated SAT assignments
- let $b_1 \leftarrow \arg\max\{S_0, S_1\}$
- recurse on F_{b_1}

runtime?

$$\le \underbrace{n}_{\#\text{recursions}} \cdot \underbrace{\#\text{samples to get } \tfrac{\varepsilon}{6n} \text{ additive error}}_{\text{poly}\left(\underbrace{\left(\frac{\varepsilon}{6n}\right)^{-1}}_{\text{approx error}} \cdot \underbrace{\left(\frac{1}{4n}\right)^{-1}}_{\text{confidence error}}\right)} \cdot \underbrace{\text{runtime of uniform generator}}_{\text{poly in } n}$$

total: $\text{poly}\left(n, \frac{1}{\varepsilon}\right)$

$$\Pr[\text{algorithm fails}] \le \sum_{\substack{\text{recursion level} \\ i=1}}^{n} \Pr[\text{estimate bad}] \le n \cdot \frac{1}{4n} \le \frac{1}{4}.$$

Works for any d.s.r. problem!

poly time (almost)-uniform-generation of solutions

$\Downarrow$

poly time approximate counting of # solns

$\Uparrow$ what about this direction?

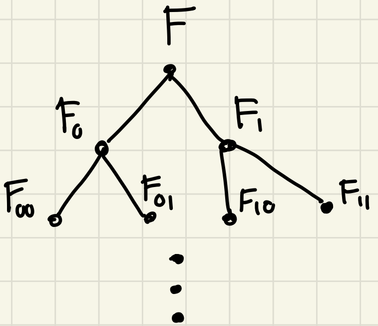
<u>Thm</u> [Jerrum Valiant Vazirani] for any problem in NP that is d.s.r.:

ptime approx counting of # solutions $\Longleftrightarrow$ ptime almost uniform generation

(easier case)

(Perfect) counting for #DNF $\Rightarrow$

(perfect) Uniform generation



Recursive algorithm:

at $b_1 \dots b_i$,

use (perfect) counter to compute

$$r_0 = \#F_{b_1 \dots b_i 0}$$

$$r_1 = F_{b_1 \dots b_i 1}$$

go left with prob $\frac{r_0}{r_0 + r_1}$

& right o.w.

<u>Claim</u> (1) always reach SAT assignment

since never take branch with 0 SAT assignments underneath

(2) $Pr[\text{output } \underbrace{b_1 \dots b_n}_{\text{SAT assignment}}] = \frac{F_{b_1}}{F} \cdot \frac{F_{b_1 b_2}}{F_{b_1}} \cdot \frac{F_{b_1 b_2 b_3}}{F_{b_1 b_2}} \cdot \dots \cdot \frac{1}{F_{b_1 b_2 \dots b_n}}$

$= \frac{1}{F}$ $\leftarrow$ same for every sat assignment

Question what if only have approx counter?

Answer $\text{RHS} \le \frac{1}{F}\left(\frac{1+\varepsilon'}{1-\varepsilon'}\right)^n \le \frac{1}{F} \cdot \frac{1}{1-\varepsilon}$

if choose $\varepsilon' < \frac{\varepsilon}{2n}$

$\Rightarrow$ close to uniform generation of sat assignments

Last time:

- poly time algorithm to uniformly generate sat assignments to DNF formula

- def of #P
  #P-complete
  FPRAS

Randomized Complexity Classes

<u>def</u>. language L is subset of $\{0,1\}^*$

e.g. $\{x \mid x$ is graph with Hamilton path$\}$

$\{x \mid x$ is collection of sets with proper 2-coloring$\}$

<u>def</u> P is class of languages L with polytime deterministic algorithm A

s.t. $x \in L \Rightarrow A(x)$ accepts

$x \notin L \Rightarrow A(x)$ rejects

def RP is class of languages L with polytime probabilistic algorithm A

s.t. $x \in L \Rightarrow \Pr[A(x) \text{ accepts}] \geq 1/2$
$x \notin L \Rightarrow \Pr[A(x) \text{ accepts}] = 0$

1-sided error

def BPP is class of languages L with polytime probabilistic algorithm A

s.t. $x \in L \Rightarrow \Pr[A(x) \text{ accepts}] \geq 2/3$
$x \notin L \Rightarrow \Pr[A(x) \text{ accepts}] \leq 1/3$

2-sided error

Comments

- constants arbitrary

  with multiplicative overhead of $O(\log 1/\beta)$
  can get error $\leq \beta$

- Clearly $P \subseteq RP \subseteq BPP$

OPEN:
is $P = BPP$?

Derandomization via Enumeration

Given: probabilistic algorithm $\mathcal{A}$ + input x

Algorithm:

Run $\mathcal{A}$ on every possible random string of length $\underbrace{r(n)}$

at most time bound of $\mathcal{A}$.

⇑

is there a better bound?

Output majority answer

Behavior:

if $x \in L$, $\geq 2/3$ of random strings cause A to accept

$\Rightarrow$ majority answer is accept

if $x \notin L$, $\geq 2/3$ of random strings cause A to reject

$\Rightarrow$ majority answer is reject

Runtime: $O(2^{r(n)} \cdot \underbrace{t(n)}_{\text{time bound of } A}) \leq O(2^{t(n)} \cdot t(n))$

note: $r(n) \leq t(n)$ but if could get a better bound on $r(n)$, would improve runtime. e.g. $r(n) = O(\log n)$ & $t(n) = \text{poly}(n)$ $\rightarrow$ total time is $\text{poly}(n)$

Corollary: $BPP \subseteq EXP$

$$EXP \equiv DTIME\left(\bigcup_c 2^{n^c}\right)$$